

UNIVERSITY OF HOUSTON

COSC 6377
Computer Networking

Dr. Omprakash Gnawali

HOMEWORK 2

Ashutosh Kumar
1854393

Introduction

In this assignment, I will create a tool that can monitor website uptime using Python language.

This monitor reads, from a `json` config file, the website to be tracked, the email to be alerted if the website is down, and the poll-interval, which tells how often we need to check the status of the website [1]. An example of the `json` config file is shown in the image below 1.



```
1 {
2   "data": [
3     {
4       "website": "https://www.apple.com/",
5       "alert": "akumar29cs@gmail.com",
6       "poll-interval": 10
7     }
8   ]
9 }
10
```

Figure 1: `json` config file

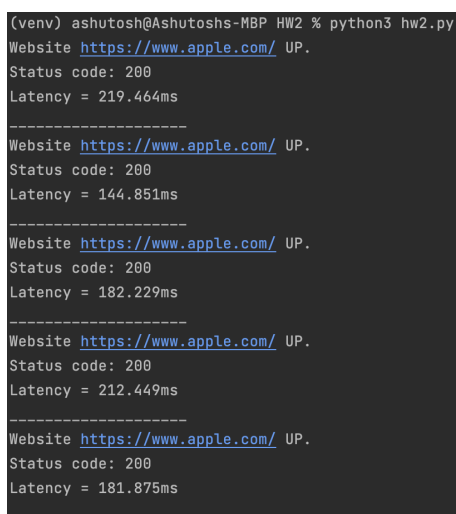
I have also created a shell script, which restarts the monitor in the case that the monitor crashes (or is shut down) [2]. My script restarts the Python monitor after it detects that the monitor is not running.

This monitor uses the Python library `smtplib` to send an email alert in case the website is down.

The various experiments performed in the assignment and their results/output are described in the following sections.

1 A website in the US

For this part of the assignment, I use my monitor to track Apple's website (<https://www.apple.com/>). When I run the monitor, the output I get can be seen in the figure 2 below.



```
(venv) ashutosh@Ashutoshs-MBP HW2 % python3 hw2.py
Website https://www.apple.com/ UP.
Status code: 200
Latency = 219.464ms

-----
Website https://www.apple.com/ UP.
Status code: 200
Latency = 144.851ms

-----
Website https://www.apple.com/ UP.
Status code: 200
Latency = 182.229ms

-----
Website https://www.apple.com/ UP.
Status code: 200
Latency = 212.449ms

-----
Website https://www.apple.com/ UP.
Status code: 200
Latency = 181.875ms

-----
```

Figure 2: Output of the monitor when tracking the uptime of <https://www.apple.com/>

1.1 Shutting down the monitor and letting the shell script restart it

Upon shutting down my monitor, the shell script automatically restarts the monitor. This can be seen in the following images 3 and 4.

The figure consists of two side-by-side terminal screenshots. The left terminal shows the output of a Python script named 'hw2.py' which monitors the website 'https://www.apple.com/'. It displays status code 200 and latency values (218.309ms, 189.471ms, 158.814ms, 138.822ms). A traceback error occurs, indicating a 'monitor()' function call. The right terminal shows the execution of a shell script './restart_monitor.sh' which responds to the crash by displaying 'hw2.py is not running. Restarting in 10 seconds...'.

Figure 3: Shell script automatically restarts the monitor when it is shut down (or it crashes)

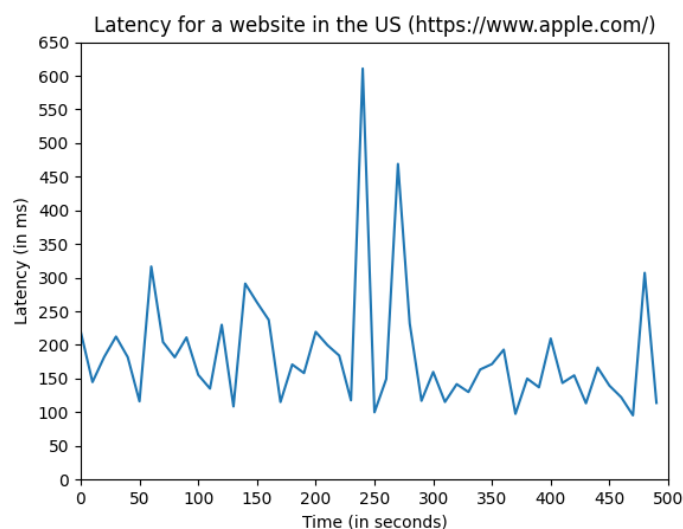
The figure is a single terminal screenshot showing the output of the shell script './restart_monitor.sh'. It displays the message 'hw2.py is not running. Restarting in 10 seconds...' followed by the successful execution of the monitor script, showing status code 200 and latency values (583.692ms, 125.551ms).

Figure 4: The monitor starts running after being restarted by the shell script

1.2 Latency Data and Graph

I let my monitor track the website for some time and collected the latency data. Some of the data can be seen in the figure 5 and the graph of the collected data can be seen in the figure 6.

timestamp	website	status_code	latency	poll_time
2023-11-11 19:32:45.345092	https://www.apple.com/	200	219.464	0
2023-11-11 19:32:55.495455	https://www.apple.com/	200	144.851	10
2023-11-11 19:33:05.683514	https://www.apple.com/	200	182.229	20
2023-11-11 19:33:15.898458	https://www.apple.com/	200	212.449	30
2023-11-11 19:33:26.083823	https://www.apple.com/	200	181.875	40
2023-11-11 19:33:36.204307	https://www.apple.com/	200	116.169	50
2023-11-11 19:33:46.523065	https://www.apple.com/	200	316.538	60
2023-11-11 19:33:56.729692	https://www.apple.com/	200	204.38	70
2023-11-11 19:34:06.918685	https://www.apple.com/	200	181.708	80
2023-11-11 19:34:17.133769	https://www.apple.com/	200	211.146	90
2023-11-11 19:34:27.295189	https://www.apple.com/	200	155.579	100
2023-11-11 19:34:37.434836	https://www.apple.com/	200	135.137	110
2023-11-11 19:34:47.670508	https://www.apple.com/	200	229.93	120
2023-11-11 19:34:57.783155	https://www.apple.com/	200	108.644	130
2023-11-11 19:35:08.077111	https://www.apple.com/	200	291.168	140
2023-11-11 19:35:18.342204	https://www.apple.com/	200	263.265	150
2023-11-11 19:35:28.584306	https://www.apple.com/	200	237.16	160
2023-11-11 19:35:38.705210	https://www.apple.com/	200	115.195	170
2023-11-11 19:35:48.877564	https://www.apple.com/	200	170.928	180
2023-11-11 19:35:59.041681	https://www.apple.com/	200	158.24	190
2023-11-11 19:36:09.268223	https://www.apple.com/	200	219.46	200

Figure 5: Some of the data collected by my monitor for **Apple's** websiteFigure 6: Latency of **Apple's** website, as tracked by my monitor

2 A slow website outside the US

For this part of the assignment, I used the website for the Indian hospital **All India Institute Of Medical Sciences (AIIMS)** (<https://www.aiims.edu/>). The output from the monitor can be seen in the following figure 7.

```

(venv) ashutosh@Ashutoshs-MBP HW2 % python3 hw2.py
Website https://www.aiims.edu/ UP.
Status code: 200
Latency = 5596.71ms
-----
Website https://www.aiims.edu/ UP.
Status code: 200
Latency = 6176.403ms
-----
Website https://www.aiims.edu/ UP.
Status code: 200
Latency = 5388.875ms
-----
Website https://www.aiims.edu/ UP.
Status code: 200
Latency = 6379.2ms
-----
Website https://www.aiims.edu/ UP.
Status code: 200
Latency = 5969.241ms
-----

```

Figure 7: Output from the monitor when tracking a slow website outside of the US

2.1 Latency Data and Graph

I let my monitor track the website for some time and collected the latency data. Some of the data can be seen in the figure 8 and the graph of the collected data can be seen in the figure 9.

timestamp	website	status_code	latency	poll_time
2023-11-11 19:44:44.966893	https://www.aiims.edu/	200	5596.71	0
2023-11-11 19:45:01.149071	https://www.aiims.edu/	200	6176.403	10
2023-11-11 19:45:16.540450	https://www.aiims.edu/	200	5388.875	20
2023-11-11 19:45:32.926080	https://www.aiims.edu/	200	6379.2	30
2023-11-11 19:45:48.897990	https://www.aiims.edu/	200	5969.241	40
2023-11-11 19:46:05.180337	https://www.aiims.edu/	200	6276.541	50
2023-11-11 19:46:19.621011	https://www.aiims.edu/	200	4438.48	60
2023-11-11 19:46:34.365007	https://www.aiims.edu/	200	4737.752	70
2023-11-11 19:46:49.109487	https://www.aiims.edu/	200	4741.057	80
2023-11-11 19:47:05.292119	https://www.aiims.edu/	200	6179.253	90
2023-11-11 19:47:21.426749	https://www.aiims.edu/	200	6132.565	100
2023-11-11 19:47:34.963190	https://www.aiims.edu/	200	3530.281	110
2023-11-11 19:47:50.377696	https://www.aiims.edu/	200	5408.258	120
2023-11-11 19:48:05.705240	https://www.aiims.edu/	200	5323.637	130
2023-11-11 19:48:21.726434	https://www.aiims.edu/	200	6018.326	140
2023-11-11 19:48:37.976436	https://www.aiims.edu/	200	6243.523	150
2023-11-11 19:48:54.555702	https://www.aiims.edu/	200	6573.36	160
2023-11-11 19:49:11.447701	https://www.aiims.edu/	200	6889.641	170
2023-11-11 19:49:26.808904	https://www.aiims.edu/	200	5355.321	180
2023-11-11 19:49:42.680424	https://www.aiims.edu/	200	5865.608	190
2023-11-11 19:49:57.710086	https://www.aiims.edu/	200	5023.621	200

Figure 8: Some of the data collected by my monitor for AIIMS' website

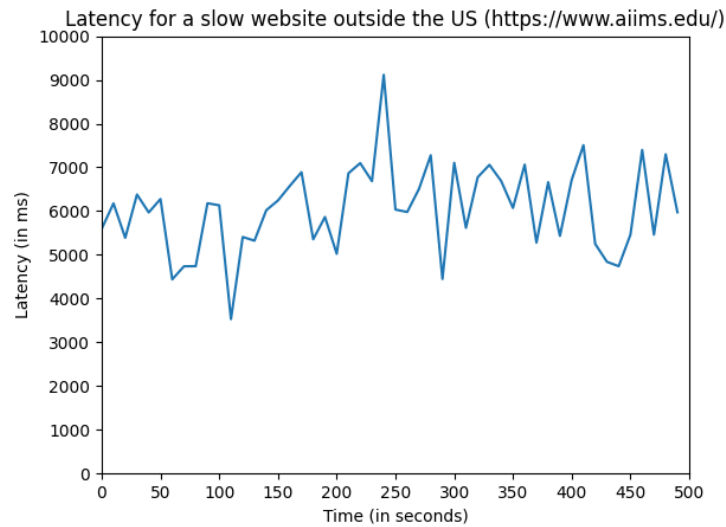


Figure 9: Latency of **AIIMS** website, as tracked by my monitor

3 My own website

3.1 How I created & hosted this website

I created a small website (just a single `html` file) and hosted it using **ngrok**. The website can be seen in the image 10 and the terminal with **ngrok** can be seen in the image 11.



Figure 10: My own website

```
ngrok
Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status
Account      online
akumar29.cs@gmail.com (Plan: Free)
Version      3.4.0
Region      United States (us)
Latency      41ms
Web Interface http://127.0.0.1:4040
Forwarding   https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app -> http://localhost:5500

Connections
  ttl  opn  rt1  rt5  p50  p90
  123   1   0.08 0.05  5.01  5.02

HTTP Requests
-----
GET /      200 OK
GET /      200 OK
GET /      200 OK
GET /      200 OK
GET /      200 OK
GET /      200 OK
GET /      502 Bad Gateway
GET /      502 Bad Gateway
GET /      200 OK
GET /      200 OK
```

Figure 11: ngrok console

3.2 Shutting the website down and testing whether the monitor sends an email alert


I ran the monitor, then shut down my website for a few seconds and then brought it back up. The results can be seen in the image below 12.

```
(venv) ashutosh@Ashutoshs-MBP HW2 % python3 hw2.py
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app UP.
Status code: 200
Latency = 222.881ms
-----
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app UP.
Status code: 200
Latency = 217.352ms
-----
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app DOWN.
Status code: 502
Latency = 342.311ms
-----
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app DOWN.
Status code: 502
Latency = 244.292ms
-----
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app UP.
Status code: 200
Latency = 218.63ms
-----
```

Figure 12: Output of the monitor with my own website

When I shut my website down, the monitor sent an email to the email specified in the config file, as seen in the figure 13 below.

Website <https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app> down! Inbox x

 **akumar29cs@gmail.com**
to akumar29cs ▾ 8:38 PM (3 minutes ago)

Hello!

The website <https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app> is down. Status code: 502.

Figure 13: Email sent by the monitor when the website is down

When I deliberately stopped the monitor, the shell script that I wrote restarts the monitor. This can be seen in the following image 14.

```
(venv) ashutosh@Ashutoshs-MBP HW2 % ./restart_monitor.sh
hw2.py is not running. Restarting in 10 seconds...
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app UP.
Status code: 200
Latency = 225.909ms
-----
Website https://4a48-2600-1700-f2f1-37e0-c99e-e7f6-a9ea-a181.ngrok-free.app UP.
Status code: 200
Latency = 699.095ms
-----
```

Figure 14: The shell script automatically restarts the monitor if it crashes.

3.3 Latency Data and Graph

I let my monitor track the website for some time and collected the latency data. Some of the data can be seen in the figure 15 and the graph of the collected data can be seen in the figure 16.

timestamp	website	status_code	latency	poll_time
2023-11-11 20:17:40.225704	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	281.933	0
2023-11-11 20:17:50.435892	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	205.162	10
2023-11-11 20:18:00.671465	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	231.134	20
2023-11-11 20:18:10.873782	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	200.844	30
2023-11-11 20:18:21.072281	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	192.344	40
2023-11-11 20:18:31.270659	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	193.54	50
2023-11-11 20:18:41.464460	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	187.355	60
2023-11-11 20:18:51.664329	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	195.635	70
2023-11-11 20:19:01.941705	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	269.732	80
2023-11-11 20:19:12.142802	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	198.455	90
2023-11-11 20:19:22.344564	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	195.595	100
2023-11-11 20:19:32.555359	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	209.579	110
2023-11-11 20:19:42.755372	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	193.717	120
2023-11-11 20:19:52.951327	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	194.991	130
2023-11-11 20:20:03.148202	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	190.489	140
2023-11-11 20:20:13.386497	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	232.407	150
2023-11-11 20:20:23.778517	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	383.729	160
2023-11-11 20:20:34.117332	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	332.793	170
2023-11-11 20:20:44.415139	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	289.888	180
2023-11-11 20:20:54.634150	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	215.461	190
2023-11-11 20:21:04.831382	https://4a48-2600-1700-f2f1-37eb-c99e-e7f6-a9ea-a181.ngrok-free.app	200	193.615	200

Figure 15: Some of the data collected by my monitor for my website

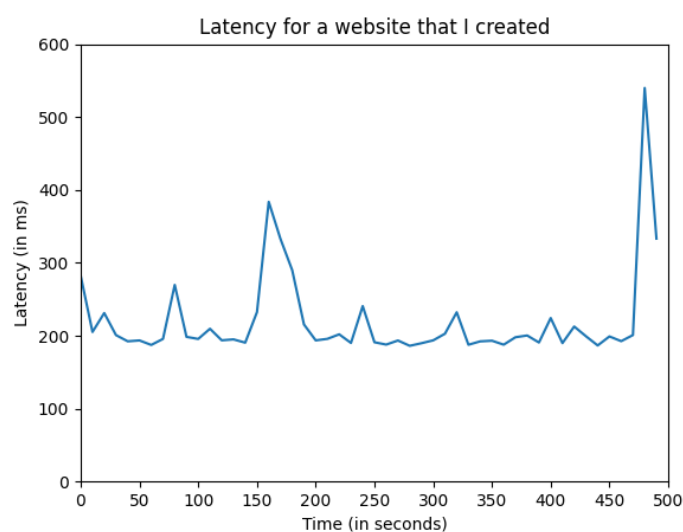


Figure 16: Latency of my website, as tracked by my monitor

4 Cost Calculations

To calculate how much this uptime monitor might cost if run on the cloud, we can use the [Amazon Web Services \(AWS\) cost calculator](#). We can use the service [AWS Lambda](#) for our monitor. For a poll interval for 10 seconds, we need to invoke the program 6 times per minute. We do not need much memory, so we can choose the lowest options available.

Assuming that the latency for a website in the US is between 200ms and 400ms, we get a cost estimate of somewhere between \$0.16 and \$0.27 per month. The calculator's calculations can be seen in the following figures [17a](#) & [17b](#).

Unit conversions
Number of requests: 6 per minute * (60 minutes in an hour x 730 hours in a month) = 262800 per month
Amount of memory allocated: 128 MB x 0.0009765625 GB in a MB = 0.125 GB
Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB
Pricing calculations
262,800 requests x 200 ms x 0.001 ms to sec conversion factor = 52,560.00 total compute (seconds)
0.125 GB x 52,560.00 seconds = 6,570.00 total compute (GB-s)
6,570.00 GB-s x 0.0000166667 USD = 0.11 USD (monthly compute charges)
262,800 requests x 0.0000002 USD = 0.05 USD (monthly request charges)
0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
0.11 USD + 0.05 USD = 0.16 USD
Lambda costs - Without Free Tier (monthly): 0.16 USD

(a) Cost calculations for a latency of 200ms

Unit conversions
Number of requests: 6 per minute * (60 minutes in an hour x 730 hours in a month) = 262800 per month
Amount of memory allocated: 128 MB x 0.0009765625 GB in a MB = 0.125 GB
Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB
Pricing calculations
262,800 requests x 400 ms x 0.001 ms to sec conversion factor = 105,120.00 total compute (seconds)
0.125 GB x 105,120.00 seconds = 13,140.00 total compute (GB-s)
13,140.00 GB-s x 0.0000166667 USD = 0.22 USD (monthly compute charges)
262,800 requests x 0.0000002 USD = 0.05 USD (monthly request charges)
0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
0.22 USD + 0.05 USD = 0.27 USD
Lambda costs - Without Free Tier (monthly): 0.27 USD

(b) Cost calculations for a latency of 400ms

Figure 17: AWS Cost Calculations to track websites with latencies of 200ms & 400ms

For the slow website that I used, the latency was quite high. Assuming a latency of 6000ms to 8000ms, we get the cost estimate of \$3.34 to \$4.43 per month. The calculations for the same can be seen in the following figures 18a & 18b.

Unit conversions
Number of requests: 6 per minute * (60 minutes in an hour x 730 hours in a month) = 262800 per month
Amount of memory allocated: 128 MB x 0.0009765625 GB in a MB = 0.125 GB
Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB
Pricing calculations
262,800 requests x 6,000 ms x 0.001 ms to sec conversion factor = 1,576,800.00 total compute (seconds)
0.125 GB x 1,576,800.00 seconds = 197,100.00 total compute (GB-s)
197,100.00 GB-s x 0.0000166667 USD = 3.29 USD (monthly compute charges)
262,800 requests x 0.0000002 USD = 0.05 USD (monthly request charges)
0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
3.29 USD + 0.05 USD = 3.34 USD
Lambda costs - Without Free Tier (monthly): 3.34 USD

(a) Cost calculations for a latency of 6000ms

Unit conversions
Number of requests: 6 per minute * (60 minutes in an hour x 730 hours in a month) = 262800 per month
Amount of memory allocated: 128 MB x 0.0009765625 GB in a MB = 0.125 GB
Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB
Pricing calculations
262,800 requests x 8,000 ms x 0.001 ms to sec conversion factor = 2,102,400.00 total compute (seconds)
0.125 GB x 2,102,400.00 seconds = 262,800.00 total compute (GB-s)
262,800.00 GB-s x 0.0000166667 USD = 4.38 USD (monthly compute charges)
262,800 requests x 0.0000002 USD = 0.05 USD (monthly request charges)
0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
4.38 USD + 0.05 USD = 4.43 USD
Lambda costs - Without Free Tier (monthly): 4.43 USD

(b) Cost calculations for a latency of 8000ms

Figure 18: AWS Cost Calculations to track websites with latencies of 6000ms & 8000ms

References

- [1] [Stack Overflow](#): How to send email alert through python if a string is found in a csv file?
- [2] [StackExchange](#): How can I make a bash script that determines if a program is already running?